Lightweight Agent Simulations (LASS)

User Manual

Version: 0.1

September, 23, 2011

Brian Krisler
bkrisler@bbn.com
Raytheon BBN Technologies

Jacob Beal
jbeal@bbn.com
Raytheon BBN Technologies

1 General Information

This is a guide to executing a demonstration of Light-weight agent based simulations using Unity 3D and Proto. This is build on top of the more general LASS framework, which allows arbitrary lightwiehgt demos to be constructed.

1.1 Overview

The Lightweight Agent Simulations (LASS) is a toolkit for conducting design-space simulations involving design / human interactions. By leveraging a game-based simulation environment, and spatial computing, designers have the ability to quickly create simulations that consider humans within the operational environment of a design, allowing the design to include design/human interaction much earlier in the design cycle.

1.2 Authorized Use Permission

The BBN LASS Toolkit is licensed to BAE Systems Inc. under the terms of the META contract. Use on other projects is not authorized. Please inquire with the contractual point of contact, listed in Section 1.3, if you wish to license the BBN LASS Toolkit for additional purposes.

1.3 Points of Contact

Technical questions relating to the installation, usage, design, or documentation of the BBN LASS Toolkit should be directed to Brian Krisler. Inqueries related to licensing and contractual matters should be directed to Dr. Jacob Beal. Contact information for these individuals is listed on the cover page of this document, or you may reach them through Raytheon BBN Technologies' main telephone number, 617-873-8000

1.4 Intended Audience

This User Manual is intended for anyone interested in seeing how LASS controlled agents interact within the Unity environment. It assumes that the reader has a computer running the OS X (10.6+) operating system.

2 System Overview

2.1 Unity 3D

Unity 3D is a state of the art game development tool, designed to allow for quick authoring of 3D scenes based upon realistic terrains and physics.

2.2 Proto

Proto is a language designed for implementing distributed systems on spatially-structured networks of static or mobile devices. The programmer describes a desired behavior for the continuous space occupied by the network, which can be translated into a program by which individual devices cooperate to produce an approximation of the desired behavior. Such programs can then be rendered into an executable than can run equivalently on different hardware platforms.

3 Getting Started

To begin using LASS, the following items are required:

- A computer running OS X 10.6+
- A copy of the LASS dmg
- A copy of the Proto dmg

3.1 Installing Proto

1. Double click on Proto.dmg
2. Navigate to the mounted volume Proto.pkg
3. Double click on the Proto.pkg.
4. Follow the steps to install Proto.

3.1 Installing The LASS Toolkit

1. Double click in the LASS dmg.
5. Drag the LASS executable to a directory such as ***/Applications***.

(*The remanider of this guide assumes the software was installed into Applications*)

4 Executing the LASS Demonstration

The LASS simulation toolkit is delivered as an OS X compiled executable application. To view and interact with the demonstration, navigate to **Applications** and double click on the application icon for LASS to launch the toolkit. Select your desired resolution and quality, and continue. Once launched, a scene is displayed containing a group of soldiers randomly placed around an MRAP vehicle (See Figure 1).

The default simulation example is based upon Proto flocking. When the agents are placed within the scene, they become aware of the agents within their network radius. (The default radius is 35). Depending upon the placement of the agents, multiple network groups of agents can exist within the scenario. When movement is triggered, all of the

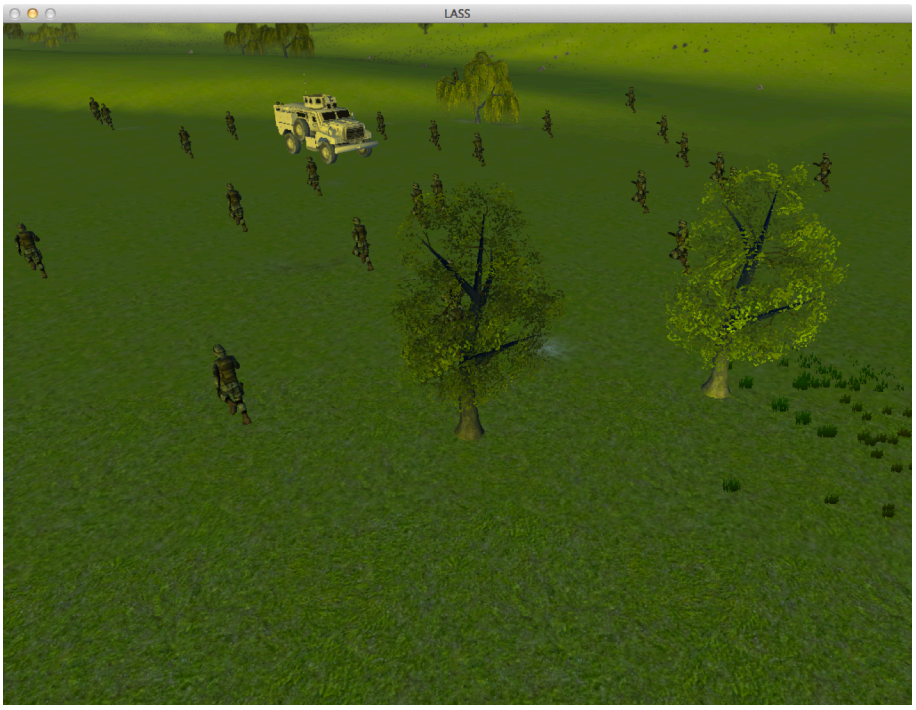

Figure 1. LASS demonstration application. The LASS demonstration application displays a group of soldiers randomly placed around an MRAP.

agents within a network group will *flock* together into a group. To see an example of flocking, press the ***M*** key. To stop movement, press the ***M*** key again.

4.1 Viewing The Simulation Demonstration

The LASS simulation toolkit provides multiple camera angles for viewing the simulation. The main viewpoints are overhead and 3rd person view. To toggle between the two main viewpoints, press the **.** (period) key while viewing the simulation.

4.1.1 3rd Person Viewpoint

Third person view mode allows the viewer to *follow* an agent as they move about the scene. The camera is placed above and slightly behind the agent, presenting a view port that includes the entire followed individual from behind (See Figure 2).When in 3rd person view, the camera can be positioned behind one of ten elected group leader AI units within the scenario. Determining which unit to follow is done by pressing the number key between **0** and **9**, that cooresponds to the unit id of the desired agent to follow. As the unit moves about the scene, the camera will continue to follow the selected unit.

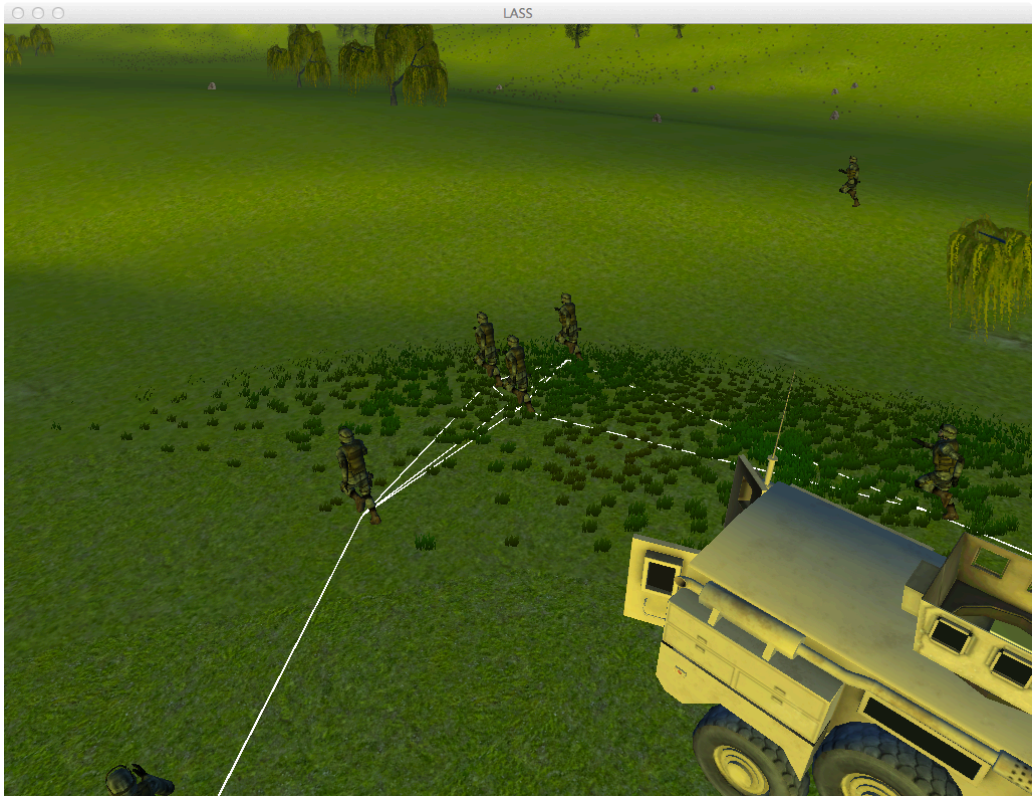

Figure 2. 3rd Person View. When in 3rd person view, the camera follows the specified AI from a distance above and behind the AI.

4.1.2 Overhead Viewpoint

The overhead viewpoint provides a birds-eye view on the scenario. This view is best for gaining a better understanding of how all the AI interact together as a cohesive unit (See Figure 3).

When in overhead mode, the camera is not attached to any specific AI unit and can be moved about the scene as desired by the user. To move the camera, the keys **A,S,W,D** are used for basic navigation.The **W** key will move the camera forward, the **S** key moves the camera backward and **A** and **S** will move the camera left and right respectivly. When in overhead mode, the operation also has the ability to zoom the camera in and out using the **[** and **]** keys.

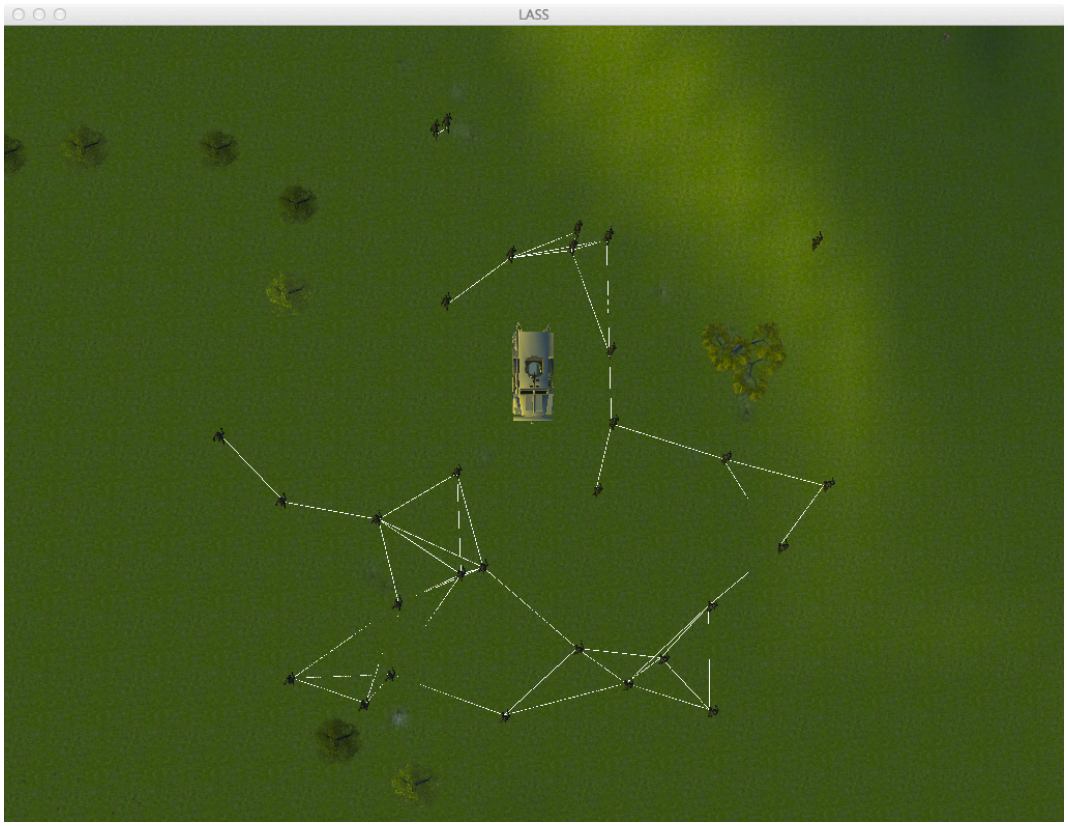

Figure 3. Overhead Viewpoint. When viewing from the overhead camera, the user can see the simulation from a birds-eye view.

4.1.3 Network Connections

When the agents interact in the simulation, they interact spatially with respect to the other agents around them. If two agents are not within range of each other, they do not influnce the actions of each other. To have a better understanding of what agents are interacting together as a group, it is useful to view the network topology between the agents. The simulation provides the controls to enable and disable the network connections between the agents. When in either the 3rd person or overhead viewpoint, pressing the **C** key will enable the network connections between the agents.

4.2 Viewing and Changing the Executed Proto Program

The LASS Toolkit allows for exploration of various proto programs within the simulation. The default program, upon launch is an simple flocking algorithm. All agents within network range will come together into a group.

The LASS Toolkit provides the ability to change the Proto program executed in the simulation. To exectue a different Proto program, press the ***U*** key to display the script editor. A UI window is displayed in the simulation that allows for modification and input

of Proto code. To apply the new code, press the update button. Upon restarting the agent movement (***M***), the new code will get executed.[1] For other programs to execute in the simulator, refer to Appendix B.

4.3 Executing the Simulation

To execute the simulation, press the **M** key. This action initiates agent movement where the agents will move about the scene based upon the logic of the Proto program. In this simulation, the Proto program executed is a form of flocking, where each network group will come together into a tight group.

During execution, pressing **M** a second time will pause the simulation. The placement of the units within the scene can also get randomized by pressing the **R** key. When the simulation is over, pressing the **Q** key will close the application.

5 Further Reading

The LASS Toolkit demonstration is based upon the Proto spatial computing language as leverages the Unity 3D game engine for the simulation environment. To learn more about Unity or Proto, please refer to the following sources:

5.1 Proto

- Project website: http://proto.bbn.com/

5.2 Unity

- Product website: http://unity3d.com/

Appendix A: Demonstration Keymap

Camera Controls	
A	Move the overhead camera left
D	Move the overhead camera right
S	Move the overhead camera back
W	Move the overhead camera forward
. [PERIOD]	Toggle between the overhead camera and the 3rd person camera
0-9	Places the 3rd person camera on agent with UID 0-9
[	Increase the camera distance

[1] The current implementation does not report compile issues. If the new code does not compile correctly, the previous program will continue to execute.

]	Decrease the camera distance

Simulator Controls	
M	Start agent movement
R	Randomly redistribute agent placement
+	Increase the agent communication radius
-	Decrease the agent communication radius

Visualization Controls	
C	Display the agent network connections
H	Display the help screen
V	Display the Unit IDs
U	Display the UI for view/editing the proto program
Q	Quit the demonstration

Appendix B: Example Proto Programs

The following programs are examples that can be used to demonstrate the effects of proto on agents within the simulation.

Flocking

The flocking program is the default program when the Toolkit is run without any modifications to the proto code. This example will cause all agents within the radius to come together into a group.

Program:

```
(all (def flock (dir) (rep v (tup 0 0 0) (let ((d (normalize (int-hood
(if (< (nbr-range) 5) (* -1 (normalize (nbr-vec))) (if (> (nbr-range)
10) (* 0.2 (normalize (nbr-vec))) (normalize (nbr v)))))))) (normalize
(+ dir (mux (> (vdot d d) 0) d v)))))) (mov (flock (tup 0 0))))
```

Vector Movement

The vector movement will case all agents within network radius of agent UID 0 to move towards the east, while all agents within the network of agent UID 1 will move towards the north. If these networks come within range of each other, all agents from both groups will turn and move together towards the northeast.

Program:

```
(let ((d0 (distance-to (= (mid) 0))) (d1 (distance-to (= (mid) 1))))
(mov (tup (< d0 (inf)) (< d1 (inf)))))
```

Move Northeast

The move northeast program is a very simple program that will result in all agents moving as a group towards the northeast.

Program:

```
(mov (tup 1 1))
```